

---

# Preconditioning

## Noisy, Ill-Conditioned Linear Systems

---

James G. Nagy  
Emory University  
Atlanta, GA

---

### Outline

---

1. The Basic Problem
  2. Regularization / Iterative Methods
  3. Preconditioning
  4. Example: Image Restoration
  5. Summary
-

## Basic Problem

Linear system of equations

$$\mathbf{b} = A\mathbf{x}$$

where

- $A$ ,  $\mathbf{b}$  are known
- $A$  is large, structured
- Goal: Compute an approximation of  $\mathbf{x}$

## Basic Problem

Linear system of equations

$$\mathbf{b} = A\mathbf{x} + \mathbf{e}$$

where

- $A$ ,  $\mathbf{b}$  are known
- $A$  is large, structured, **ill-conditioned**
- Goal: Compute an approximation of  $\mathbf{x}$

## Basic Problem

Linear system of equations

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e}$$

where

- $A$ ,  $\mathbf{b}$  are known
- $A$  is large, structured, **ill-conditioned**
- Goal: Compute an approximation of  $\mathbf{x}$

---

**Applications:** Ill-posed inverse problems.

- Geomagnetic Prospecting
- Tomography
- Image Restoration

## Basic Problem

Linear system of equations

$$\mathbf{b} = A\mathbf{x} + \mathbf{e}$$

where

- $A$ ,  $\mathbf{b}$  are known
- $A$  is large, structured, **ill-conditioned**
- Goal: Compute an approximation of  $\mathbf{x}$

---

**Applications:** Ill-posed inverse problems.

- Geomagnetic Prospecting
- Tomography
- **Image Restoration**

$\mathbf{b}$  = observed image

$A$  = blurring matrix (structured)

$\mathbf{e}$  = noise

$\mathbf{x}$  = true image

## Basic Problem

Linear system of equations

$$\mathbf{b} = A\mathbf{x} + \mathbf{e}$$

where

- $A$ ,  $\mathbf{b}$  are known
- $A$  is large, structured, **ill-conditioned**
- Goal: Compute an approximation of  $\mathbf{x}$

---

**Applications:** Ill-posed inverse problems.

- Geomagnetic Prospecting
- Tomography
- **Image Restoration**

$\mathbf{b}$  = observed image

$A$  = blurring matrix (structured)

$\mathbf{e}$  = noise

$\mathbf{x}$  = true image

Example →

## Basic Problem – Properties

### Computational difficulties revealed through SVD:

Let  $A = U\Sigma V^T$  where

- $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_N)$ ,  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_N \geq 0$
- $U^T U = I$ ,  $V^T V = I$

## Basic Problem – Properties

### Computational difficulties revealed through SVD:

Let  $A = U\Sigma V^T$  where

- $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_N)$ ,  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_N \geq 0$
  - $U^T U = I$ ,  $V^T V = I$
- 

For ill-posed inverse problems,

- $\sigma_1 \approx 1$ , small singular values cluster at 0
- small singular values  $\Rightarrow$  oscillating singular vectors

## Basic Problem – Properties

Inverse solution for noisy, ill-posed problems:

If  $A = U\Sigma V^T$ , then

$$\begin{aligned}\mathbf{x} &= A^{-1}\mathbf{b} \\ &= V\Sigma^{-1}U^T\mathbf{b} \\ &= \sum_{i=1}^n \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i\end{aligned}$$

## Basic Problem – Properties

Inverse solution for noisy, ill-posed problems:

If  $A = U\Sigma V^T$ , then

$$\begin{aligned}\hat{\mathbf{x}} &= A^{-1}(\mathbf{b} + \mathbf{e}) \\ &= V\Sigma^{-1}U^T(\mathbf{b} + \mathbf{e}) \\ &= \sum_{i=1}^n \frac{\mathbf{u}_i^T(\mathbf{b} + \mathbf{e})}{\sigma_i} \mathbf{v}_i\end{aligned}$$

## Basic Problem – Properties

Inverse solution for noisy, ill-posed problems:

If  $A = U\Sigma V^T$ , then

$$\begin{aligned}\hat{\mathbf{x}} &= A^{-1}(\mathbf{b} + \mathbf{e}) \\ &= V\Sigma^{-1}U^T(\mathbf{b} + \mathbf{e}) \\ &= \sum_{i=1}^n \frac{\mathbf{u}_i^T(\mathbf{b} + \mathbf{e})}{\sigma_i} \mathbf{v}_i \\ &= \sum_{i=1}^n \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i + \sum_{i=1}^n \frac{\mathbf{u}_i^T \mathbf{e}}{\sigma_i} \mathbf{v}_i \\ &= \mathbf{x} + \text{error}\end{aligned}$$

## Basic Problem – Properties

Inverse solution for noisy, ill-posed problems:

If  $A = U\Sigma V^T$ , then

$$\begin{aligned}\hat{\mathbf{x}} &= A^{-1}(\mathbf{b} + \mathbf{e}) \\ &= V\Sigma^{-1}U^T(\mathbf{b} + \mathbf{e}) \\ &= \sum_{i=1}^n \frac{\mathbf{u}_i^T(\mathbf{b} + \mathbf{e})}{\sigma_i} \mathbf{v}_i \\ &= \sum_{i=1}^n \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i + \sum_{i=1}^n \frac{\mathbf{u}_i^T \mathbf{e}}{\sigma_i} \mathbf{v}_i \\ &= \mathbf{x} + \text{error}\end{aligned}$$

Example →

## Regularization

**Basic Idea:** Filter out effects of small singular values.

$$\mathbf{x}_{\text{reg}} = \sum_{i=1}^n \phi_i \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i$$

where the "filter factors" satisfy

$$\phi_i \approx \begin{cases} 1 & \text{if } \sigma_i \text{ is large} \\ 0 & \text{if } \sigma_i \text{ is small} \end{cases}$$

# Regularization

Some regularization methods:

1. Truncated SVD

$$\mathbf{x}_{\text{tsvd}} = \sum_{i=1}^k \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i$$

2. Tikhonov

$$\mathbf{x}_{\text{tik}} = \sum_{i=1}^n \frac{\sigma_i^2}{\sigma_i^2 + \alpha^2} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i$$

3. Wiener

$$\mathbf{x}_{\text{wien}} = \sum_{i=1}^n \frac{\delta_i \sigma_i^2}{\delta_i \sigma_i^2 + \gamma_i^2} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i$$

## Iterative Regularization

### Basic idea:

- Use an iterative method (e.g., conjugate gradients)
- Terminate iteration before theoretical convergence:
  - Early iterations reconstruct solution.
  - Later iterations reconstruct noise.

# Iterative Regularization

## Basic idea:

- Use an iterative method (e.g., conjugate gradients)
  - Terminate iteration before theoretical convergence:
    - Early iterations reconstruct solution.
    - Later iterations reconstruct noise.
- 

## Some important methods:

- CGLS, LSQR, GMRES
- MR2 (Hanke, '95)
- MRNSD (Kaufman, '93; N., Strakos, '00)

## Iterative Regularization

Efficient for large problems, provided

1. Multiplication with  $A$  is not expensive.
2. Convergence is rapid enough.

## Iterative Regularization

Efficient for large problems, provided

1. Multiplication with  $A$  is not expensive.

Image restoration  $\Leftrightarrow$  Use FFTs

2. Convergence is rapid enough.
  - CGLS, LSQR, GMRES, MR2 often fast, especially for severely ill-posed, noisy problems.
  - MRNSD based on steepest descent, typically converges very slowly.

## Iterative Regularization

Efficient for large problems, provided

1. Multiplication with  $A$  is not expensive.

Image restoration  $\Leftrightarrow$  Use FFTs

2. Convergence is rapid enough.
  - CGLS, LSQR, GMRES, MR2 often fast, especially for severely ill-posed, noisy problems.
  - MRNSD based on steepest descent, typically converges very slowly.

Example  $\rightarrow$

## Preconditioning

Purposes of preconditioning:

1. Accelerate convergence.

- Apply iterative method to  $P^{-1}Ax = P^{-1}b$ .

## Preconditioning

Purposes of preconditioning:

1. Accelerate convergence.

- Apply iterative method to  $P^{-1}Ax = P^{-1}b$ .
- In this case we minimize  $\|x\|_2$ .

2. Enforce regularization constraint on solution.  
(Hanke, '92; Hansen, '98)

- Apply iterative method to  $AL^{-1}Lx = b$ .
- In this case, we minimize  $\|Lx\|_2$ .

## Preconditioning for Regularization

Basic idea:

- Find a matrix  $L$  to enforce smoothness constraint

$$\min \|Lx\|_2$$

- Typically  $L$  approximates a derivative operator.

## Preconditioning for Speed

Typical approach for  $A\mathbf{x} = \mathbf{b}$

- Find matrix  $P$  so that  $P^{-1}A \approx I$ .

- "Ideal" choice:  $P = A$

In this case, converge in one iteration to  $\mathbf{x} = A^{-1}\mathbf{b}$

## Preconditioning for Speed

Typical approach for  $A\mathbf{x} = \mathbf{b}$

- Find matrix  $P$  so that  $P^{-1}A \approx I$ .

- "Ideal" choice:  $P = A$

In this case, converge in one iteration to  $\mathbf{x} = A^{-1}\mathbf{b}$

---

For ill-conditioned, noisy problems

- Inverse solution is corrupted with noise

## Preconditioning for Speed

Typical approach for  $A\mathbf{x} = \mathbf{b}$

- Find matrix  $P$  so that  $P^{-1}A \approx I$ .
  - "Ideal" choice:  $P = A$   
In this case, converge in one iteration to  $\mathbf{x} = A^{-1}\mathbf{b}$
- 

For ill-conditioned, noisy problems

- Inverse solution is corrupted with noise
- "Ideal" regularized preconditioner: If  $A = U\Sigma V^T$   
(Hanke, N., Plemmons, '93)

$$P = U\Sigma_k V^T = U\text{diag}(\sigma_1, \dots, \sigma_k, \mathbf{1}, \dots, \mathbf{1})V^T$$

## Preconditioning for Speed

Notice that the preconditioned system is:

$$\begin{aligned} P^{-1}A &= (U\Sigma_k V^T)^{-1}(U\Sigma V^T) \\ &= V\Sigma_k^{-1}\Sigma V^T \\ &= V\Delta V^T \end{aligned}$$

where  $\Delta = \text{diag}(1, \dots, 1, \sigma_{k+1}, \dots, \sigma_n)$

That is,

- Large (**good**) singular values clustered at 1.
- Small (**bad**) singular values not clustered.

## Preconditioning for Speed

Remaining questions:

1. How to choose truncation index,  $k$ ?

Use regularization parameter choice methods,  
e.g., GCV, L-curve, Picard condition

2. We can't compute SVD, so now what?

Use SVD approximation.

## Preconditioning for Speed

An SVD Approximation:

- Decompose  $A$  as: (Van Loan and Pitsianis, '93)

$$A = C_1 \otimes D_1 + C_2 \otimes D_2 + \cdots + C_k \otimes D_k$$

where  $C_1 \otimes D_1 = \operatorname{argmin} \|A - C \otimes D\|_F$ .

## Preconditioning for Speed

An SVD Approximation:

- Decompose  $A$  as: (Van Loan and Pitsianis, '93)

$$A = C_1 \otimes D_1 + C_2 \otimes D_2 + \cdots + C_k \otimes D_k$$

where  $C_1 \otimes D_1 = \operatorname{argmin} \|A - C \otimes D\|_F$ .

- Choose a “structured” (or sparse)  $U$  and  $V$ .

## Preconditioning for Speed

An SVD Approximation:

- Decompose  $A$  as: (Van Loan and Pitsianis, '93)

$$A = C_1 \otimes D_1 + C_2 \otimes D_2 + \cdots + C_k \otimes D_k$$

where  $C_1 \otimes D_1 = \operatorname{argmin} \|A - C \otimes D\|_F$ .

- Choose a “structured” (or sparse)  $U$  and  $V$ .
- Let  $\Sigma = \operatorname{argmin} \|A - U\Sigma V^T\|_F$ .

## Preconditioning for Speed

An SVD Approximation:

- Decompose  $A$  as: (Van Loan and Pitsianis, '93)

$$A = C_1 \otimes D_1 + C_2 \otimes D_2 + \cdots + C_k \otimes D_k$$

where  $C_1 \otimes D_1 = \operatorname{argmin} \|A - C \otimes D\|_F$ .

- Choose a “structured” (or sparse)  $U$  and  $V$ .
- Let  $\Sigma = \operatorname{argmin} \|A - U\Sigma V^T\|_F$ .

That is,

$$\Sigma = \operatorname{diag}(U^T A V)$$

## Preconditioning for Speed

An SVD Approximation:

- Decompose  $A$  as: (Van Loan and Pitsianis, '93)

$$A = C_1 \otimes D_1 + C_2 \otimes D_2 + \cdots + C_k \otimes D_k$$

where  $C_1 \otimes D_1 = \operatorname{argmin} \|A - C \otimes D\|_F$ .

- Choose a “structured” (or sparse)  $U$  and  $V$ .
- Let  $\Sigma = \operatorname{argmin} \|A - U\Sigma V^T\|_F$ .

That is,

$$\begin{aligned}\Sigma &= \operatorname{diag} (U^T A V) \\ &= \operatorname{diag} \left( U^T \left( \sum_{i=1}^k C_i \otimes D_i \right) V \right)\end{aligned}$$

## Preconditioning for Speed

Choices for  $U$  and  $V$  depend on problem (application).

- Since

$$A = C_1 \otimes D_1 + C_2 \otimes D_2 + \cdots + C_k \otimes D_k$$

and

$$C_1 \otimes D_1 = \operatorname{argmin} \|A - C \otimes D\|_F$$

we might use singular vectors of  $C_1 \otimes D_1$ .

## Preconditioning for Speed

Choices for  $U$  and  $V$  depend on problem (application).

- Since

$$A = C_1 \otimes D_1 + C_2 \otimes D_2 + \cdots + C_k \otimes D_k$$

and

$$C_1 \otimes D_1 = \operatorname{argmin} \|A - C \otimes D\|_F$$

we might use singular vectors of  $C_1 \otimes D_1$ .

- For image restoration, we also use

Fourier Transforms (FFTs)

Discrete Cosine Transforms (DCTs)

## Example: Image Restoration

1. Matrix Structure
2. Efficiently computing SVD approximation

## Matrix Structure in Image Restoration

First, how do we get the matrix,  $A$ ?

## Matrix Structure in Image Restoration

First, how do we get the matrix,  $A$ ?

- Using linear algebra notation, the  $i$ -th column of  $A$  can be written as:

$$Ae_i = \begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_i & \cdots & \mathbf{a}_n \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \mathbf{a}_i$$

## Matrix Structure in Image Restoration

First, how do we get the matrix,  $A$ ?

- Using linear algebra notation, the  $i$ -th column of  $A$  can be written as:

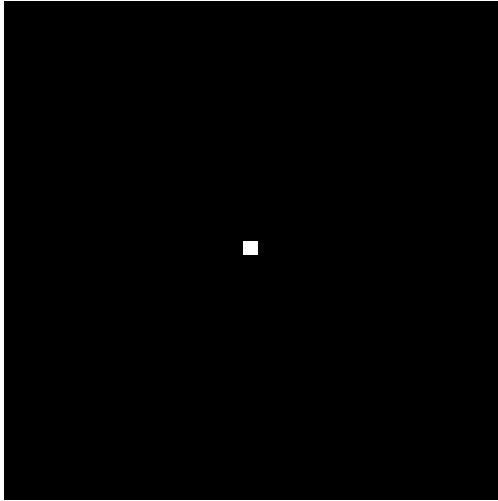
$$Ae_i = \begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_i & \cdots & \mathbf{a}_n \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \mathbf{a}_i$$

- In an imaging system,

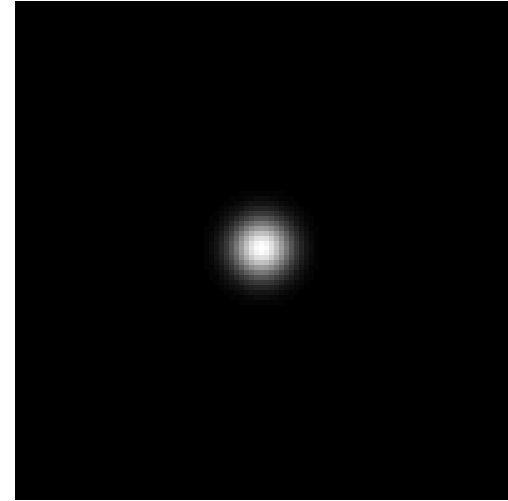
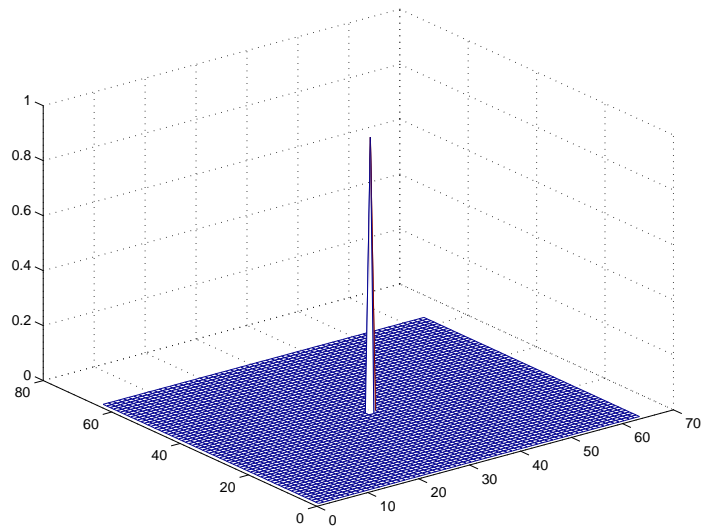
$e_i$  = point source

$Ae_i$  = point spread function (PSF)

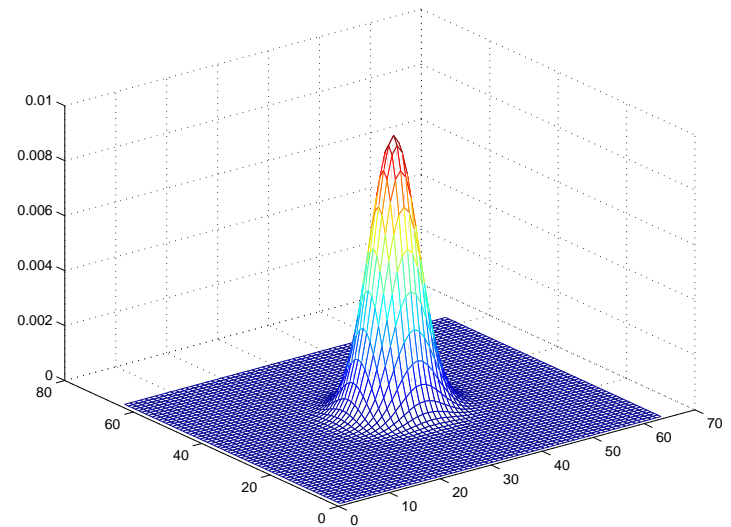
# Matrix Structure in Image Restoration



point source

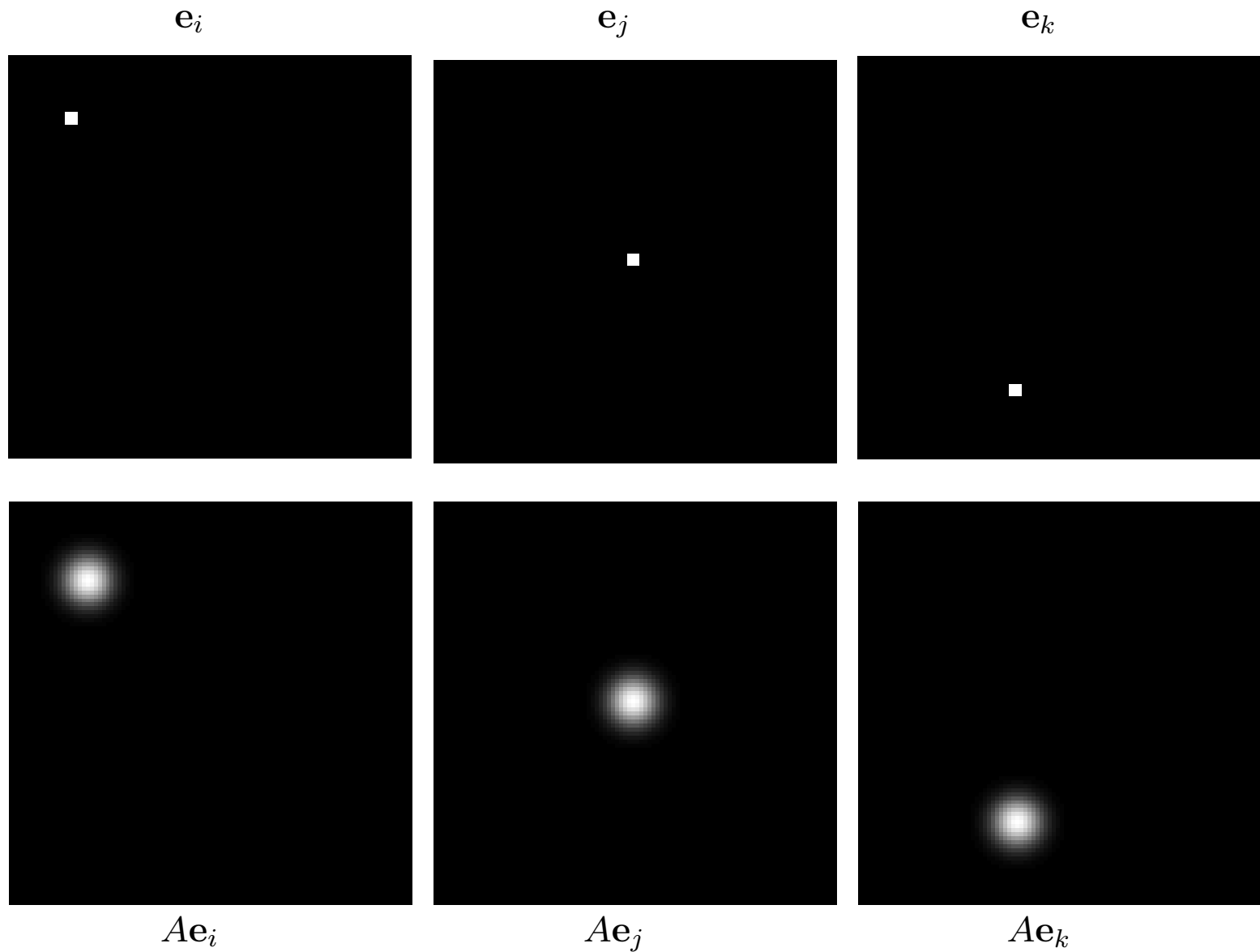


PSF



# Matrix Structure in Image Restoration

Spatially invariant PSF implies:



## Matrix Structure in Image Restoration

That is, spatially invariant implies

- Each column of  $A$  is identical, modulo shift.
- One point PSF is enough to fully describe  $A$ .
- $A$  has Toeplitz structure.

# Matrix Structure in Image Restoration

$$\mathbf{e}_5 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \xrightarrow{\text{blur}} \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \rightarrow A\mathbf{e}_5 = \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{31} \\ p_{32} \\ p_{33} \end{bmatrix}$$

$$A = \begin{bmatrix} & & p_{11} & & \\ & & p_{12} & & \\ & & p_{13} & & \\ & & p_{21} & & \\ & & p_{22} & & \\ & & p_{23} & & \\ & & p_{31} & & \\ & & p_{32} & & \\ & & p_{33} & & \end{bmatrix}$$



# Matrix Structure in Image Restoration

$$\mathbf{e}_5 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \xrightarrow{\text{blur}} \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \rightarrow A\mathbf{e}_5 = \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{31} \\ p_{32} \\ p_{33} \end{bmatrix}$$

$$A = \left[ \begin{array}{ccc|cc} p_{22} & p_{21} & & p_{12} & p_{11} \\ p_{23} & p_{22} & p_{21} & p_{13} & p_{12} & p_{11} \\ & p_{23} & p_{22} & & p_{13} & p_{12} \\ \hline p_{32} & p_{31} & & p_{22} & p_{21} & & p_{12} & p_{11} \\ p_{33} & p_{32} & p_{31} & p_{23} & p_{22} & p_{21} & p_{13} & p_{12} & p_{11} \\ & p_{33} & p_{32} & & p_{23} & p_{22} & & p_{13} & p_{12} \\ \hline & & & p_{32} & p_{31} & & p_{22} & p_{21} \\ & & & p_{33} & p_{32} & p_{31} & p_{23} & p_{22} & p_{21} \\ & & & & p_{33} & p_{32} & & p_{23} & p_{22} \end{array} \right]$$

# Matrix Structure in Image Restoration

## Matrix Summary

Boundary Condition	Matrix Structure
zero	BTTB
periodic	BCCB
reflexive	BTTB+BTHB+BHTB+BHHB

B = block  
T = Toeplitz  
C = circulant  
H = Hankel

# Matrix Structure in Image Restoration

## Matrix Summary

Boundary Condition	Matrix Structure
zero	BTTB
periodic	BCCB (use FFT)
reflexive (strongly symmetric)	BTTB+BTHB+BHTB+BHHB (use DCT)

B = block  
T = Toeplitz  
C = circulant  
H = Hankel

## Matrix Structure in Image Restoration

For a separable PSF, we get:

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \mathbf{c}\mathbf{d}^T = \begin{bmatrix} c_1d_1 & c_1d_2 & c_1d_3 \\ c_2d_1 & c_2d_2 & c_2d_3 \\ c_3d_1 & c_3d_2 & c_3d_3 \end{bmatrix} \rightarrow \mathbf{A}\mathbf{e}_5 = \begin{bmatrix} c_1 \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \\ c_2 \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \\ c_3 \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \end{bmatrix}$$

## Matrix Structure in Image Restoration

For a separable PSF, we get:

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \mathbf{c}\mathbf{d}^T = \begin{bmatrix} c_1d_1 & c_1d_2 & c_1d_3 \\ c_2d_1 & c_2d_2 & c_2d_3 \\ c_3d_1 & c_3d_2 & c_3d_3 \end{bmatrix} \rightarrow \mathbf{A}\mathbf{e}_5 = \begin{bmatrix} c_1 \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \\ c_2 \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \\ c_3 \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \end{bmatrix}$$

$$\begin{bmatrix} & c_1 \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} & \\ \hline & c_2 \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} & \\ \hline & c_3 \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} & \\ \hline \end{bmatrix}$$

# Matrix Structure in Image Restoration

For a separable PSF, we get:

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \mathbf{c}\mathbf{d}^T = \begin{bmatrix} c_1d_1 & c_1d_2 & c_1d_3 \\ c_2d_1 & c_2d_2 & c_2d_3 \\ c_3d_1 & c_3d_2 & c_3d_3 \end{bmatrix} \rightarrow \mathbf{A}\mathbf{e}_5 = \begin{bmatrix} c_1 \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \\ c_2 \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \\ c_3 \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \left| \begin{array}{c} c_1 \begin{pmatrix} d_2 & d_1 \\ d_3 & d_2 \\ & d_3 & d_2 \end{pmatrix} \end{array} \right| \\ \left| \begin{array}{c} c_2 \begin{pmatrix} d_2 & d_1 \\ d_3 & d_2 \\ & d_3 & d_2 \end{pmatrix} \end{array} \right| \\ \left| \begin{array}{c} c_3 \begin{pmatrix} d_2 & d_1 \\ d_3 & d_2 \\ & d_3 & d_2 \end{pmatrix} \end{array} \right| \end{bmatrix}$$

# Matrix Structure in Image Restoration

For a separable PSF, we get:

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \mathbf{c}\mathbf{d}^T = \begin{bmatrix} c_1d_1 & c_1d_2 & c_1d_3 \\ c_2d_1 & c_2d_2 & c_2d_3 \\ c_3d_1 & c_3d_2 & c_3d_3 \end{bmatrix} \rightarrow \mathbf{Ae}_5 = \begin{bmatrix} c_1 \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \\ c_2 \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \\ c_3 \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \end{bmatrix}$$

$$\left[ \begin{array}{c|c|c} c_2 \begin{pmatrix} d_2 & d_1 \\ d_3 & d_2 \\ d_3 & d_2 \end{pmatrix} & c_1 \begin{pmatrix} d_2 & d_1 \\ d_3 & d_2 \\ d_3 & d_2 \end{pmatrix} & \\ \hline c_3 \begin{pmatrix} d_2 & d_1 \\ d_3 & d_2 \\ d_3 & d_2 \end{pmatrix} & c_2 \begin{pmatrix} d_2 & d_1 \\ d_3 & d_2 \\ d_3 & d_2 \end{pmatrix} & c_1 \begin{pmatrix} d_2 & d_1 \\ d_3 & d_2 \\ d_3 & d_2 \end{pmatrix} \\ \hline & c_3 \begin{pmatrix} d_2 & d_1 \\ d_3 & d_2 \\ d_3 & d_2 \end{pmatrix} & c_2 \begin{pmatrix} d_2 & d_1 \\ d_3 & d_2 \\ d_3 & d_2 \end{pmatrix} \end{array} \right] = \mathbf{C} \otimes \mathbf{D}$$

## Matrix Structure in Image Restoration

If the PSF is not separable, we can still compute:

$$P = \sum_{i=1}^r \mathbf{c}_i \mathbf{d}_i^T \quad (\text{sum of rank-1 matrices})$$

and therefore, get

$$A = \sum_{i=1}^r C_i \otimes D_i \quad (\text{sum of Kron. products})$$

In fact, we can get “optimal” decompositions.

(Kamm, N, '00; N., Ng, Perrone, 03)

## SVD Approximation for Image Restoration

Use  $A \approx U\Sigma V^T$ , where

- If  $\mathcal{F}(\sum C_i \otimes D_i) \mathcal{F}^*$  is best,

$$U = V = \mathcal{F}^*, \quad \Sigma = \text{diag} \left( \mathcal{F} \left( \sum C_i \otimes D_i \right) \mathcal{F}^* \right)$$

- If  $\mathcal{C}(\sum C_i \otimes D_i) \mathcal{C}^T$  is best,

$$U = V = \mathcal{C}^T, \quad \Sigma = \text{diag} \left( \mathcal{C} \left( \sum C_i \otimes D_i \right) \mathcal{C}^T \right)$$

- If  $(U_c \otimes U_d)^T (\sum C_i \otimes D_i) (V_c \otimes V_d)$  is best,

$$U = U_c \otimes U_d, \quad V = V_c \otimes V_d,$$

$$\Sigma = \text{diag} \left( (U_c \otimes U_d)^T \left( \sum C_i \otimes D_i \right) (V_c \otimes V_d) \right)$$

Example →

## The End

- Preconditioning ill-posed problems is difficult, but possible.
- Can build approximate SVD from Kronecker product approximations.
- Can implement efficiently for image restoration.
- Matlab software: [RestoreTools](#) (Lee, N., Perrone, '02)

Object oriented approach for image restoration.

<http://www.mathcs.emory.edu/~nagy/RestoreTools/>

Related software for ill-posed problems

(Hansen, Jacobsen)

<http://www.imm.dtu.dk/~pch/Regutools/>